FF-AUTOMATION

User Manual

v.1.01

Using DB2SMS with Web Studio

Updated 6th Jun 2008

Description:

This manual describes how to use DB2SMS with Indusoft Web Studio (IWS) SCADA. DB2SMS –program is GSM/SMS –driver, which allows Web Studio to communicate with AutoLog GSM-RTUs bi-directionally. This manual should be used with IWS Demo project, which is made with IWS v.6.1. The purpose of this document with DEMO project is to show all essential features of using AutoLog GSM-RTUs with Web Studio so system designers can get as fast getting started process as possible.

PostgreSQL database is used just as interface and not for permanent storing of measurement data. Measurement data is read to Indusoft Web Studio and stored to another database using IWS's ODBC task. Also other database types can be used, but it needs some modifications in ODBC task sheets.

FF-Automation would appreciate to get any feedback of this manual and DEMO project.

Any questions and feedback can be sent to antti.moijanen@ff-automation.com

1 Table of contents

1 TABLE OF CONTENTS 2

2 GENERAL 3

2.1 ABOUT THE MANUAL 3
2.2 REQUIRED SOFTWARE COMPONENTS 3

3 CREATING DATABASES AND DATABASE CONNECTIONS 4

3.1 SETTING UP POSTGRESQL 4
3.1.1 Creating new user 4
3.1.2 Creating interface database 4
3.1.3 History database 4
3.2 CREATING DSNS FOR DATABASES 5
3.2.1 Creating DSN for interface database called SMSIndusoft 5

4 DESCRIPTION OF FUNCTIONALITY 7

4.1 DB2SMS (SHORT DESCRIPTION OF THE DRIVER) 7
4.2 MYTEST (SHORT DESCRIPTION OF DEMO IWS PROJECT) 8
4.2.1 User Interface 8
4.2.2 Temperature Alarm 8
4.3 DIAGRAM OF PROGRAM FUNCTONALITY 9

5 USING MYTEST -USER INTERFACE 10

5.1 PHONE NUMBERS 10
5.2 SENDING SMS –MESSAGES FROM IWS 10
5.2.1 Sending group message 10
5.2.2 Sending message to defined number 10
5.3 READING THE ERROR STATUS (DB2SMS PROGRAM) 10
5.4 RECEIVING SMS -MESSAGES 11
5.4.1 Event -messages 11
5.4.2 History -messages 11
5.4.3 Report -messages 11
5.4.4 Creating variables and history tables using Variable Configuration –tool 12
5.5 DELETING MESSAGES 14
5.6 SENDING ERROR MESSAGES 14

6 HOW THE IWS LINKS TO DB2SMS? 15

6.1 SENDING SMS -MESSAGE 15
6.2 READING ERRORMESSAGE 15
6.3 READING INCOMING MESSAGES FROM THE INTERFACE DATABASE 15
6.3.1 Phases 15
6.3.2 ODBC tasks 16
6.3.3 Scheduler tasks 17
6.3.4 Math tasks 19
6.3.5 Report and history maths (1. phase only) 20
6.4 REMOVING OLD (FLAG = 0) MESSAGES 20
6.5 WATCHDOG TIMER 21

7 TAGS AND CLASS TAGS 22

8 KNOWN PROBLEMS 25

2 General

2.1 About the manual

The manual is divided to 4 different sections:

1. Configuring databases and database connections using ODBC (Open Database Connectivity).
2. The description of functionality
3. How to use SMS features in IWS
4. Detailed description of IWS to DB2SMS connection.

This manual is focusing how to configure and use databases and How to configure Web Studio when using DB2SMS driver. The format of SMS –messages and detailed description of PostgreSQL database can be read from DB2SMS –Manual. Both manuals should be read to understand how to setup the complete system. It is preferable to start from the DB2SMS –manual.

It is suggested that the system designer reads this manual with simultaneous IWS Demo project, which is send with this manual.

This manual is not covering the description of the Demo project's operator interface or additional task like Alarm task, input scalings or alarm forwarding to GSM phone. So it is strongly suggested that System designed study carefully how the DEMO project is made. When DEMO project and the functionality of DB2SMS driver is fully understand, system designer can start developing own additional features.

2.2 Required software components.

- Indusoft Webstudio 6.1 + SP2
- DB2SMS 2.1 (supplied with DEMO project and manual)
  - .Net Framework 2.0 (http://www.microsoft.com/downloads/details.aspx?familyid=0856eacb-4362-4b0d-8edd-aab15c5e04f5&displaylang=en)
  - GSM-modem
- PostgreSQL
  - PostgreSQL (http://www.postgresql.org/ftp/binary/v8.2.3/win32/)

3 Creating databases and database connections

3.1 Setting up PostgreSQL

After you've installed PostgreSQL, you have to configure it. We recommend using *psql*-tool (Start->Programs->PostgreSQL 8.2-> psql to 'postgres') for this. When you start the program, it asks a password you've selected while installing PostgreSQL.

3.1.1 Creating new user

First you should create a new user, which is used by DB2SMS and IWS. Next command creates user *iws* with password *secure_password*:

```
CREATE ROLE iws LOGIN PASSWORD 'secure_password' NOINHERIT VALID
UNTIL 'infinity';
```

3.1.2 Creating interface database

Interface database is used as an interface between IWS and DB2SMS. Next command creates the database and sets its owner to iws.

```
CREATE DATABASE smsindusoft WITH ENCODING='SOL  ASCII' OWNER=iws:
```

A database may have multiple schemas, but we are using only one in the interface database. Next command modifies *search_path*-parameter of the new database:

```
ALTER DATABASE smsindusoft SET search  path=smsindusoft:
```

Database creation is completed, but the tables are missing. The tables may be loaded from a backup file. Open command prompt (Start->Programs->PostgreSQL 8.2-> Command Prompt) and write next command:

```
pg_restore.exe -i -h localhost -p 5432 -U postgres -d smsindusoft -v "C:\
DB2SMS\1.1\Databases\postgresql.backup"
```

You may ignore the errors given by the program.

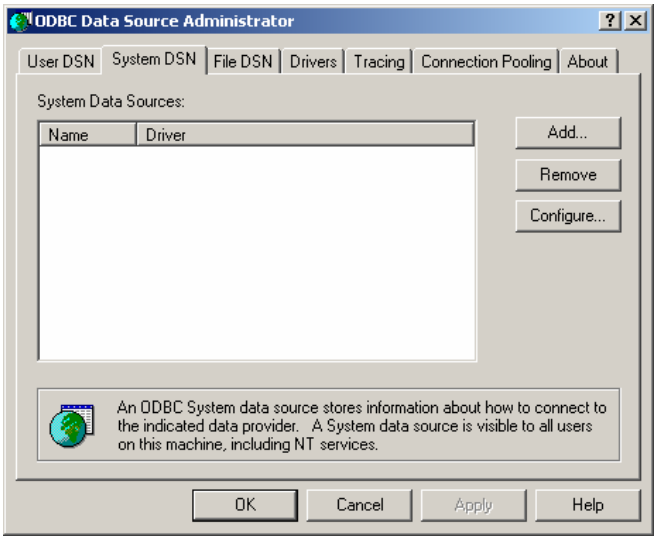
3.1.3 History database

History database will be created later while configuring variables.

3.2 Creating DSNs for databases

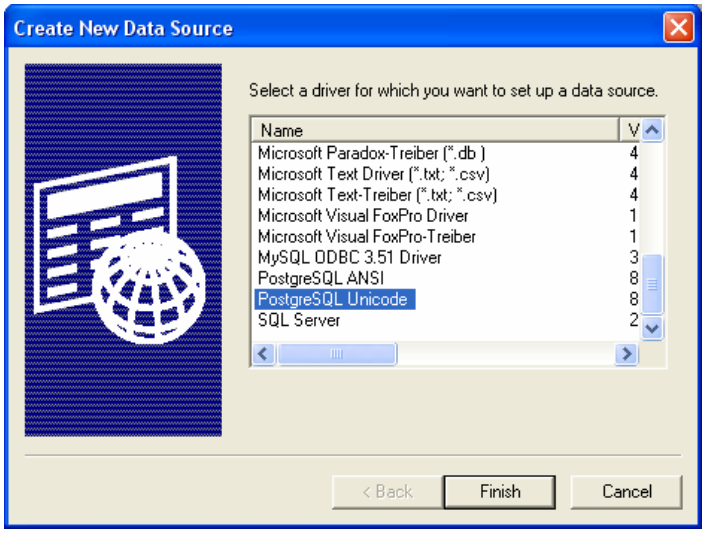
IWS uses ODBC tasks to connect with PostgreSQL databases. Here is guide how to make DNS configurations for these databases. PostgreSQL is used both to store measurement data and as an interface database between IWS and DB2SMS. Interface database is used also to store phone numbers, variable names, etc.

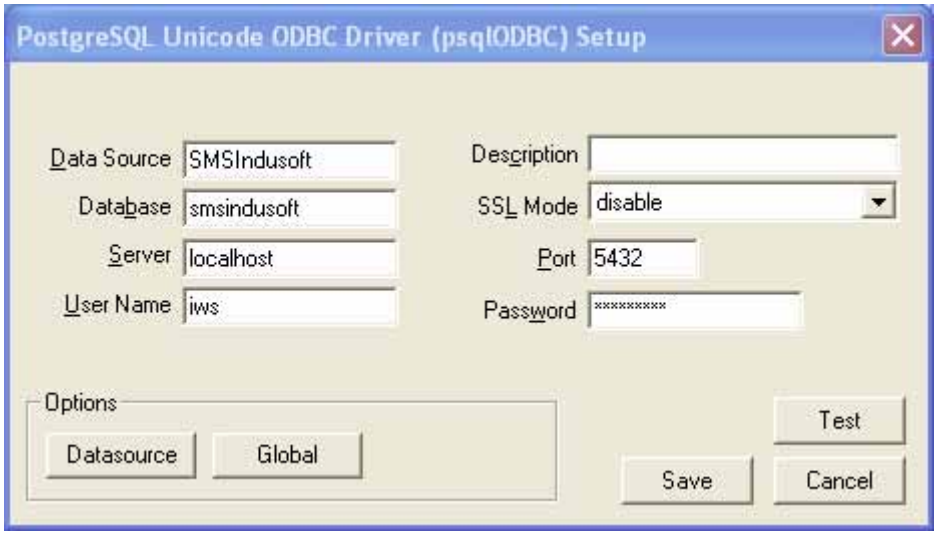
1. Select *Control Panel ->Administrative Tools ->Data Sources (ODBC) ->System DSN*



3.2.1 Creating DSN for interface database called SMSIndusoft

1. Click add..
2. select *PostgreSQL Unicode*
3. *Click Finish.*





4. A new window opens. Write *SMSIndusoft* to *Data Source*-field. Make sure you write this correctly without space. DSN (Data Source Name) is used to link IWS and the database.

   Write *smsindusoft* to *Database*-field.

   If you're using both PostgreSQL and IWS in the same computer, write *localhost* to *Server*-field. Otherwise you should write the IP-address of the server to this field.

   Write username *iws* to *User Name*-field and the password you've selected to *Password*-field.

   You may test your connection by pressing *Test*-button. When you're done, press *Save*-button.

5. Repeat steps 2-4, but use DSN Name SMSHistory instead of SMSIndusoft.
6. Click *OK.*

4 Description of functionality

DB2SMS DEMO project is a simple example of linking DB2SMS (driver) with Web Studio (IWS).

It is suggested to read DB2SMS and Indusoft Web Studio manuals to fully understand the functionality.

4.1 DB2SMS (short description of the driver)

DB2SMS is SMS –driver, which handles the outgoing and incoming SMS messages. It is responsible for “low level” communication with GSM modem, filtering the incoming SMS messages and “parsing” the messages. DB2SMS uses an interface – database to interface with other programs (like IWS).

DB2SMS supports three kinds of incoming (GSM-RTU->IWS) SMS message types:

1. *Report* –message can be used to send the latest values of GSM-RTU’s variables/Inputs. One message can have many variables. One value / variable.
2. *History* –message can be used to send many time-stamped values of one variable.
3. *Event* –messages can be used to send other kinds of text messages e.g. alarms or events.

DB2SMS supports three kinds of outgoing (IWS->GSM-RTU) SMS message types:

1. Send Group SMS –message (Same message is send to all defined GSM numbers)
2. Send one SMS –message (Message is send to one defined GSM number)
3. send System Error SMS -message. If the DB2SMS detects that IWS application is not running (watchdog pulse from IWS is not changing), it can send System Error SMS –message.

DB2SMS is supplied with DEMO project’s DB2SMS -sub folder.

4.2 MyTest (short description of DEMO IWS project)

4.2.1 User Interface

MyTest DEMO –project's user interface includes versatile features for settings configuration values in the interface -database.

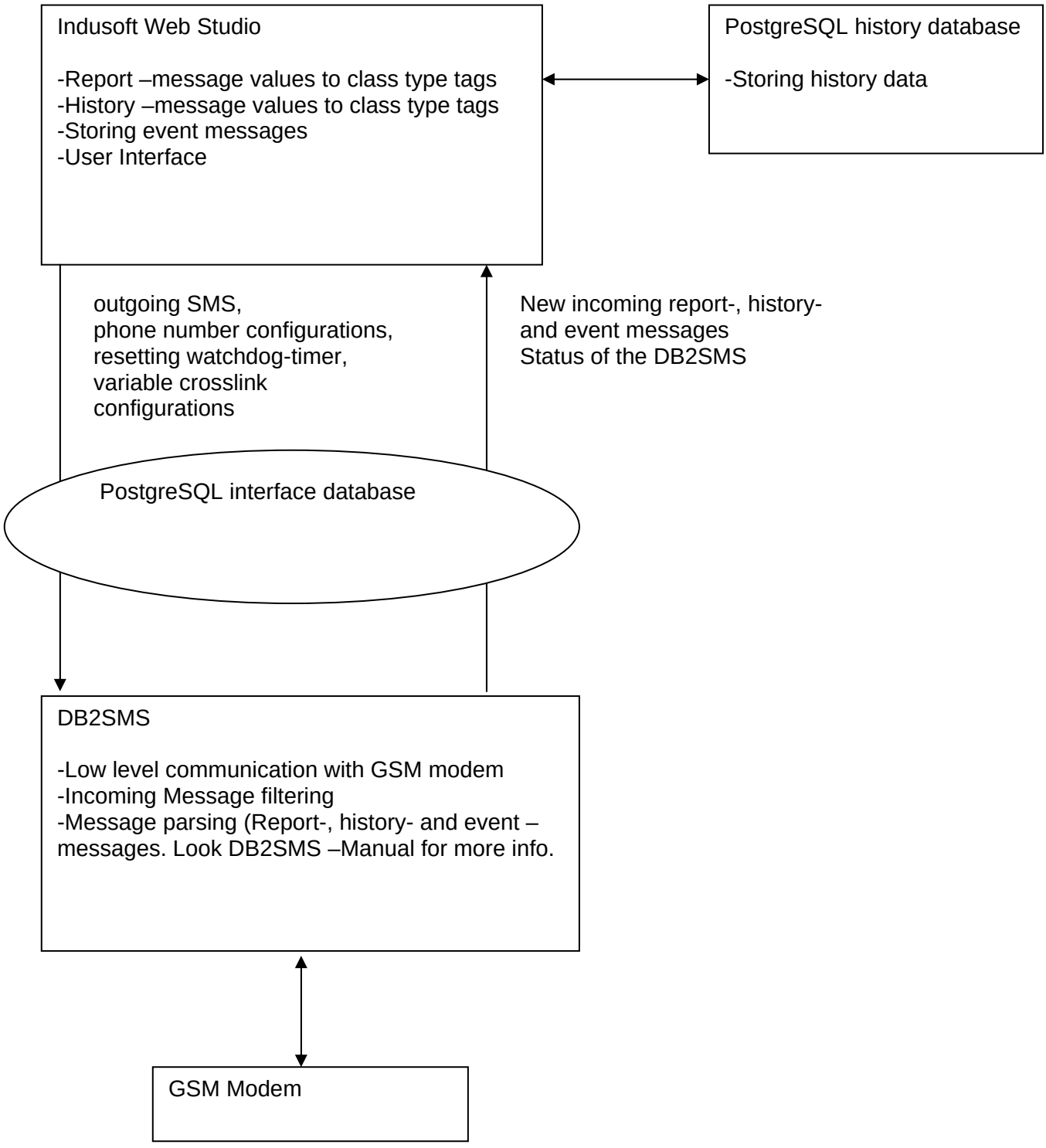
There is a ready-made trend object, that can be used to show the measurement values from the measurement history database.

User interface demonstrates also how to send SMS.

4.2.2 Temperature Alarm

MyTest application is sending alarm message to all GSM numbers in the Phonebook_Sending table if the PLC[1].Temperature value goes above 40°C. PLC is configured to send Report message every hour. PLC[x].Temperature is scaled from the PLC[x].AI1 variable (0…4096 to -50...150) using scheduler task. Alarm is created using Alarm –task.

4.3 Diagram of program functonality



5 Using MyTest -User Interface

5.1 Phone numbers

The interface database has two phone number lists:
Phonebook_Sending –table includes numbers where the Group SMS messages are send.
Phonebook_Receiving –table includes numbers which are allowed to send incoming messages.

DB2SMS copies the value from Phonebook_Receiving | Name –field to Received_ * | Name –field.

* = Report, History or Events

Only those phone numbers are active in phonebook_* tables, which Enabled –field is <>0.

5.2 Sending SMS –messages from IWS

Sending SMS –message from the user interface is quite simple. User just writes the values to flag and message tags and triggers the saving tag.

5.2.1 Sending group message

1. Write DB_SendAll.Flag = 1
2. Write to tag DB_SendAll.Message the wanted SMS message
3. Trigger the interface database ODBC write procedure. Changing the value of DBC_SendAll.Insert does this. (DBC_SendAll.Insert = NOT DBC_SendAll.Insert)

Group SMS is sent to all active GSM numbers in the Phonebook_Sending –table.

5.2.2 Sending message to defined number

1. Write DB_SendSingle.Flag = 1
2. Write to tag DB_SendSingle.Phone the wanted phone number
3. Write to tag DB_SendSingle.Message the wanted message
4. Trigger the interface database ODBC write procedure. This is done by changing the value of DBC_SendSingle.Insert (DBC_SendSingle.Insert = NOT DBC_SendSingle.Insert)

5.3 Reading the error status (DB2SMS program)

You must trigger the DBC_ErrorMessage.Select –tag to read the status of the DB2SMS –program. The Error status –message can be read from the DB_ErrorMessage -tag. This is done automatically in every

5.4 Receiving SMS -messages

5.4.1 Event -messages

When IWS detects that new Event –messages have arrived to the interface database (Received_Events | Flag = 1) it just generates IWS's SendEvent –function. SendEvent –function stores the event to the event database and it can be seen using the Event object. If the event –message starts with word "ALARM" it will be send using group SMS sending to all phone numbers that are configured in Phonebook_sending –table.

5.4.2 History -messages

When IWS detects that new History –messages have arrived to the interface database (Received_History | Flag = 1) it writes the value to the history database. IWS selects the history table name using values from sender's *Name* (Received_History | Name) and *Variable* (Received_History | Variable) fields. Table name is format trend_{Name}_{variable} e.g. trend_PLC_AI0.

ODBC –task is not creating tables to the history database so user has to create those manually. See 5.4.4

5.4.3 Report -messages

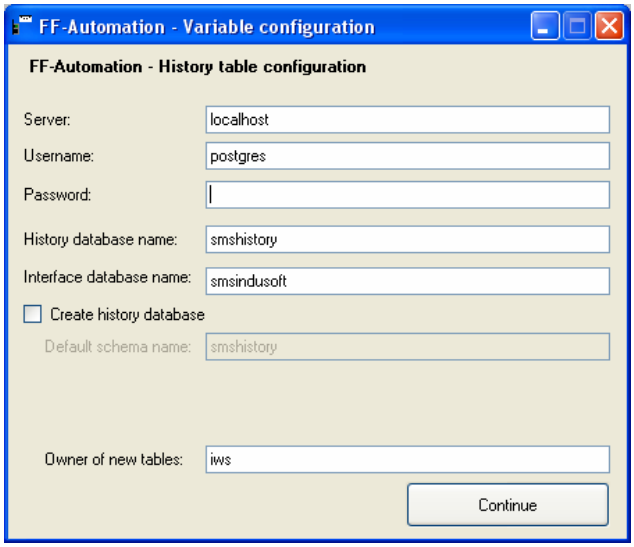
When IWS detects that new Report –messages have arrived to the interface database (Received_report | Flag = 1) it stores the value (Received_report | Value) to class variable named Received_report | Name.Received_report | Variable. E.g if sender's name is PLC and variable's name is WM0, then value is stored to class variable PLC.WM0. Note that PLC is defined in IWS to be a class variable that has many members, one of which is WM0.

Report message can have many variables but messages are parsed to the interface database so that one row in the Received_report –table has only one variable.

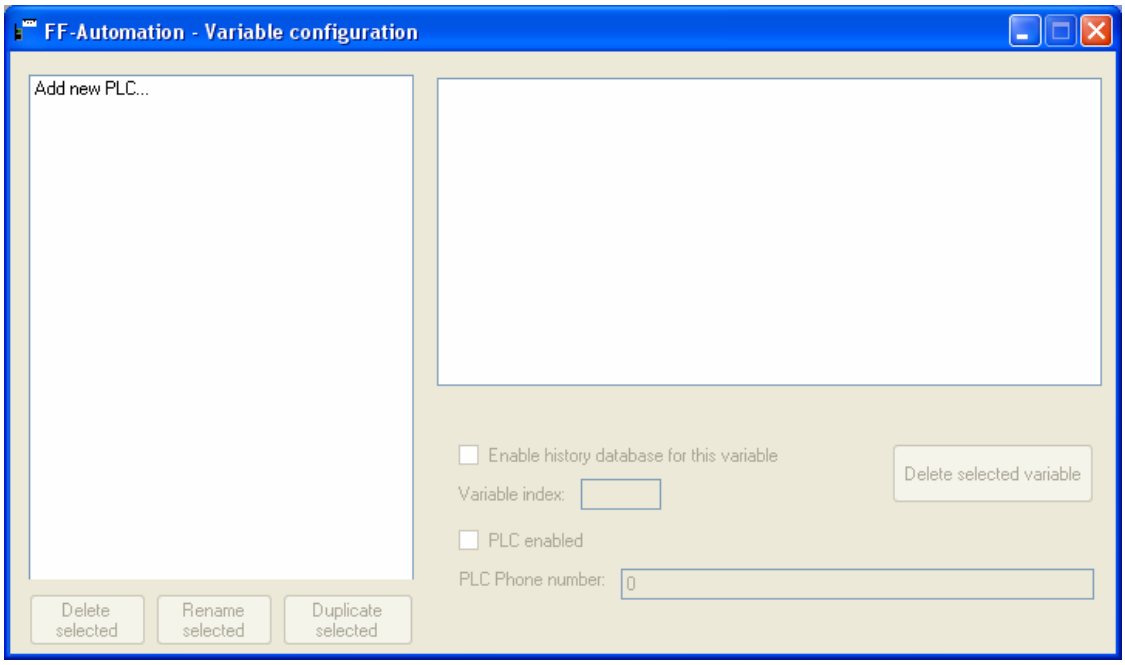
5.4.4 Creating variables and history tables with *Variable Configuration* –tool

Variable cross-linking table, receivers phonebook and history table may be created by using Variable Configuration tool, which is included with the project (IWSDemo-directory\DB2SMS\DBConfiguration.exe). Please note that the changes happen in real time. If you do a mistake, you cannot cancel!

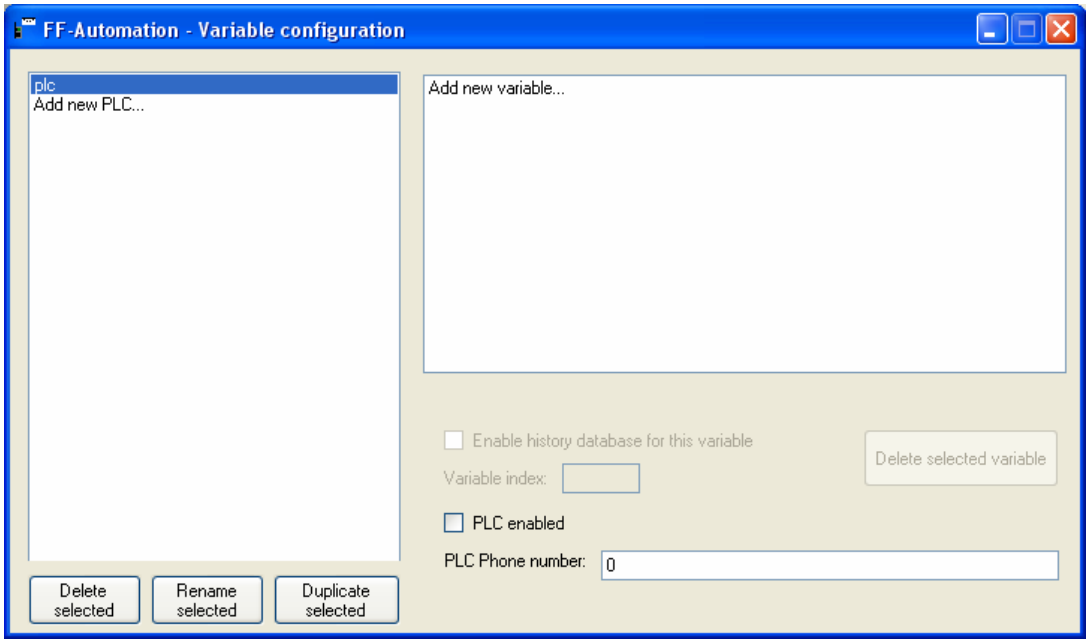
1. Start the program. You should see a login dialog:



2. Write password for user postgres. If you don't have a history database yet, you may create it by checking checkbox *Create history database*. Normally you don't need to change *Default schema name* value Click *Continue* when you're done.
3. After a while, the program opens the main window:



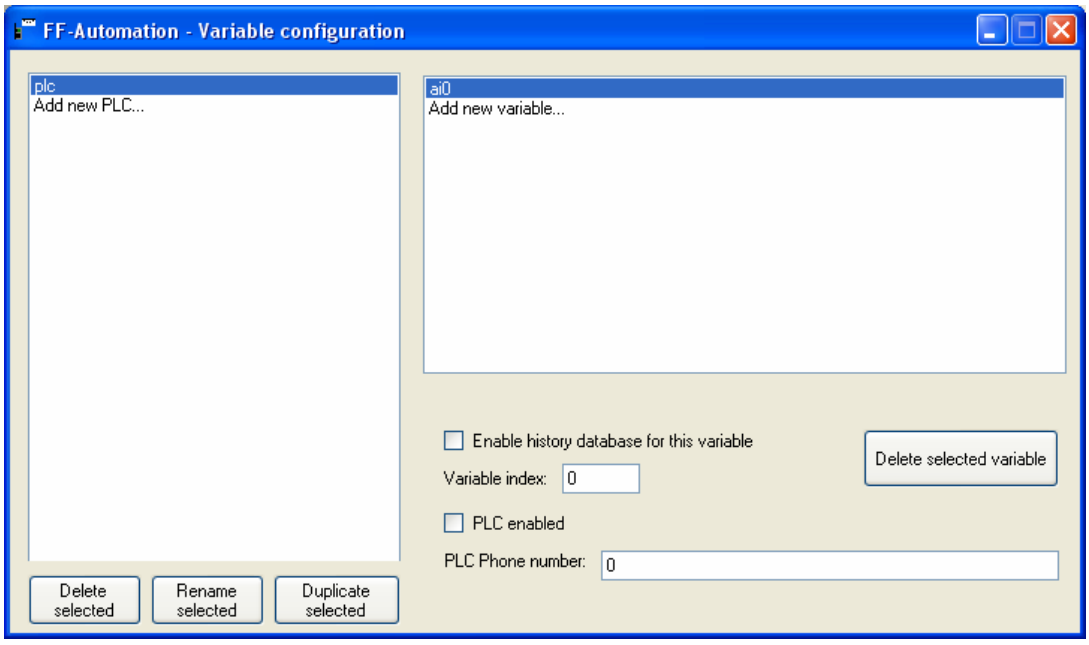
4. By double clicking *Add new PLC…* you may create a new PLC.



5. You may change PLC properties by selecting it from the list. You can remove it (*Delete selected*), rename it (*Rename selected*) or create duplicates of it (*Duplicate selected*). Please note that duplicating creates also new history tables.

   You can also change the phone number of the PLC or enable/disable it. By double clicking *Add new variable…* you may create a new variable for the selected PLC.

6. After you have created a new variable and selected it, you may create a history database for it. Just check *Enable history database for this variable* – checkbox. *Variable index* defines the index of the selected variable while receiving history or report messages.



7. When you are done with creating variables, just close the program.

5.5 Deleting messages

MyTest –demo project deletes old (flag =0) messages from the interface database – tables ~24 hours period. You can delete the old messages from the table by toggling the following tag values. (Toggling means that 1 turns to 0 or 0 turns to 1. Toggling doesn't mean that 1 turns to 2.)
:

- DBC_ReceiveEvent.Delete
- DBC_ReceiveReport.Delete
- DBC_ReceiveHistory.Delete
- DBC_SendAll.Delete
- DBC_SendSingle.Delete

Program is designed so that it will automatically select the deleted data using one minute period (Look ODBC task).

5.6 Sending error messages

IWS changes the value of the interface database's *Send_Watchdog* –table every 8 seconds. If DB2SMS detects that this value is not changing in defined time, it sends the defined SMS message to the defined phone number. From user interface you can change the error message and the phone number. You may also disable the error message sending.

6 How the IWS links to DB2SMS?

This chapter describes how the communication interface between IWS and the interface database is created. IWS uses ODBC –task to communicate with the interface database. To make the communication automatic, IWS uses scheduler and math tasks. Normally scheduler starts the certain communication process using e.g. time triggering. Math tasks are used to make more complicated scripts. Be sure that you fully understand the functionality of these scripts before you make any changes.

6.1 Sending SMS -message

This project demonstrates only the simple SMS sending process and it works only manually from the user interface. See chapter 5.2 for more information.

6.2 Reading ErrorMessage

ErrorMessage-table is read automatically once per second using Scheduler –task #1.

Clock		00:00:01	DBC_ErrorMessage.Select	NOT DBC_ErrorMessage.Select

This will start the ODBC –task called “Read DB2SMS's state from database”. The ErrorMessage is stored to tag *DB_ErrorMessage.*

6.3 Reading incoming messages from the interface database

Reading incoming SMS messages from the interface database is divided to three phases. This process reads all events (which flag =1). After first event is read it writes its flag to zero. Then it continues to read next event (which flag =1). When all the events are read, all the event flags are zero. Note that events are not deleted from the interface database after reading. Deleting events is a separated process.

Reading Events, reports and history is about similar process.

6.3.1 Phases

- Beginning (*x_Receiver.Begin_MTrig*)
  - In the beginning all the flag=1 arrays are selected from the database. Some tags are initialized.
- 1. phase (*x_Receiver.State0_MTrig*)
  - The event is read and SendEvent –function stores event to the event database. Alarm events are sent to GSM phones.
- 2. phase (*x_Receiver.State1_MTrig*)
  - The events flag is turned to zero.
- 3. phase (*x_Receiver.State2_MTrig*)
  - The next array is selected from the database and the process goes back top phase 1.
- Finish (*x_Receiver.Finish_MTrig*)
  - The Finish process just ends the event receiving process.

6.3.2 ODBC tasks

There are 17 ODBC tasks in the MyTest program:

Task name	Table	Condition	Description
1. Send SMS messages to everyone	Send_All		Send SMS messages to everyone
2. Send single SMS Messages	Send_One		Send single SMS Messages
21. Receive an event message	Received_Events	Flag=1	Receive an event messages
4. The phonebook for sending	Phonebook_Sending	Name={Send_Phones_Name}	Modifying phonebook (group messages)
6. Modify the event table	Received_Events	mindex = {DB_ReceiveEvent.Index}	Modify the event table. Modifies the event's flag to zero after reading.
7. Receive a history messages	Received_History	Flag=1	Receive a history messages
8. Modify a history table	Received_History	mindex = {DB_ReceiveHistory.Index}	Modify the history table. Modifies the history array's flag back to zero after reading.
9. Save a history column to the history db	Trend_{sender}_{var}		History data storing to thehistory database. Note that the DSN is WS_Hispool, not SMSIndusoft.
10. Receive report messages	Received_Report	Flag=1	Receive report messages
11. Modify report table	Received_Report	mindex = {DB_ReceiveReport.Index}	Modify the report table. Modifies the report array's flag back to zero after reading.
13. Read DB2SMS's state from database	ErrorMessage		Read DB2SMS's state from database
14. Remove old history messages	Received_History	Index={Crosslink_ID}	Remove old history messages
15. Remove old event messages	Received_Event	Index={Crosslink_ID}	Remove old event messages
16. Remove old report messages	Received_Report	Index={Crosslink_ID}	Remove old report messages
17. Remove old send_all messages	Send_All	Index={Crosslink_ID}	Remove old send_all messages
18. Remove old send_single messages	Send_One	Index={Crosslink_ID}	Remove old send_single messages
19. Watchdog alarm resetting	Send_Watchdog	"Name"='IWS' AND "Enabled"=1	Watchdog alarm resetting. (Watchdog should be active and name should be IWS)

6.3.3 Scheduler tasks

For every message type, there is own receiver scheduler –task. These tasks are about identical for all message types, only the tag names differs. This table represents the scheduler task, which controls the event receiving process.

Clock		00:00:05	Event_Receiver.Begin_Mtrig	1	Event_Receiver.Active=1
Change	DBC_ReceiveHistory.Transaction		Event_Receiver.State0_Mtrig	(DBC_ReceiveEvent.Status0=0) AND (Event_Receiver.State=0) AND (Event_Receiver.State2_MTrig = 0) AND Event_Receiver.Begin_MTrig = 0) AND (NOT (DB_ReceiveEvent.Phone = ""))	Event_Receiver.Active=0
Change	DBC_ReceiveHistory.MTransaction		Event_Receiver.State1_Mtrig	(DBC_ReceiveEvent.Status0=0) AND (DBC_ReceiveEvent.Status1=0) AND (Event_Receiver.State=1) AND (Event_Receiver.State0_MTrig = 0)	Event_Receiver.Active=0
Change	DBC_ReceiveHistory.MTransaction		Event_Receiver.State2_Mtrig	(DBC_ReceiveEvent.Status0=0) AND (DBC_ReceiveEvent.Status1=0) AND (Event_Receiver.State=2) AND (Event_Receiver.State1_MTrig = 0)	Event_Receiver.Active=0
Change	DBC_ReceiveHistory.Transaction		Event_Receiver.Finish_Mtrig	(NOT (DBC_ReceiveEvent.Status0=0)) OR (NOT (DBC_ReceiveEvent.Status1=0)) OR (DB_ReceiveEvent.Time="")	Event_Receiver.Active=0

First row activates the initialization. ODBC task is triggered from math function. Second row activates the saving process of the read event to the history database. This phase is activated only is there are no errors in ODBC read (*Event_Receiver.Status0=0*) and program is in the first phase (*Event_Receiver.State=0*) and third phase is not active (*Event_Receiver.State2_MTrig = 0 and initialization is not active* (*Event_Receiver.Begin_MTrig = 0*) and the read event array was not empty (*NOT (DB_ReceiveEvent.Phone = "")*).

3. and 4. rows are similar to 2. row.

The last row activates the Finishing process if something went wrong *NOT (DBC_ReceiveEvent.Status0=0)) OR (NOT (DBC_ReceiveEvent.Status1=0))*) or there are no more new events (*DB_ReceiveEvent.Time=""*).

6.3.4 Math tasks

There are five Math tasks per message type. This represents the Events Math tasks, but Report and history maths are quite similar and only phase 1 differs

6.3.4.1 Initialization (*Event_Receiver.Begin_MTrig*)

Event_Receiver.Active	1	Event receiving flag is turned active
DB_ReceiveEvent.Phone	""	Clearing phone number
DBC_ReceiveEvent.Status0	0	Clearing ODBC error status
DBC_ReceiveEvent.Status1	0	Clearing ODBC error status
Event_Receiver.State0_MTrig	0	Deactivates phase 1
Event_Receiver.State1_MTrig	0	Deactivates phase 2
Event_Receiver.State2_MTrig	0	Deactivates phase 3
Event_Receiver.State	0	Activates the first state
DBC_ReceiveEvent.Select	NOT DBC_ReceiveEvent.Select	Stars ODBC task, Selects all events, which flag = 1
Event_Receiver.Begin_MTrig	0	This math will not be executed again

6.3.4.2 Phase 1 (*Event_Receiver.State0_MTrig*)

	SendEvent(DB_ReceiveEvent.Message + " (" + DB_ReceiveEvent.Date + "/" + DB_ReceiveEvent.Time + ", " + DB_ReceiveEvent.Phone + " (" + DB_ReceiveEvent.Name + "))")	Stores the event to the history database
DB_SendAll.Message	DB_ReceiveEvent.Phone + " (" + DB_ReceiveEvent.Name +"): " + DB_ReceiveEvent.Message	Copies the event message, for group SMS purposes
DB_SendAll.Flag	1	Group message flag is set to 1
DBC_SendAll.Insert	If(StrLeft(StrLower(DB_ReceiveEvent.Message),5)="alarm",NOT DBC_SendAll.Insert,DBC_SendAll.Insert)	If the event message starts with the word alarm or ALARM, then it will be send using group SMS sending.
DBC_ReceiveEvent.Mselect	NOT DBC_ReceiveEvent.Mselect	Selects array for modification.
Event_Receiver.State	1	Moves to phase 2
Event_Receiver.State0_MTrig	0	This math will not be executed again

6.3.4.3 Phase 2 (*Event_Receiver.State1_MTrig*)

DB_ReceiveEvent.Flag	0	Flag is turned to zero (array is read)
DBC_ReceiveEvent.MUpdate	NOT DBC_ReceiveEvent.MUpdate	The flag field is updated to zero.
Event_Receiver.State	2	Next phase
Event_Receiver.State1_MTrig	0	This math will not be executed again

6.3.4.4 Phase 3 (Event_Receiver.State2.Mtrig)

DB_ReceiveEvent.Phone	""	Phone number is cleared
DBC_ReceiveEvent.Next	NOT DBC_ReceiveEvent.Next	Next event is read
Event_Receiver.State	0	Back to phase 1
Event_Receiver.State2_MTrig	0	This math will not be executed again

6.3.4.5 Finish (*Event_Receiver.Finish_MTrig*)

Event_Receiver.Active	0	Receiving has Finished
Event_Receiver.Finish_MTrig	0	This math will not be executed again

6.3.5 Report and history maths (1. phase only)

6.3.5.1 History receiving math 1. Phase (*History_Receiver.State0_MTrig*)

DB_HistoryPool.Variable	DB_ReceiveHistory.Name + "_" + DB_ReceiveHistory.Variable	This sets the correct table name in the MySQL database
DB_HistoryPool.TimeStamp	DB_ReceiveHistory.Date + " " + DB_ReceiveHistory.Time	This sets the timestamp
DB_HistoryPool.Value	DB_ReceiveHistory.Value	This sets the value
DBC_HistoryPool.Insert	NOT DBC_HistoryPool.Insert	The data is inserted to MySQL database
DBC_ReceiveHistory.Mselect	NOT DBC_ReceiveHistory.Mselect	The history array is selected for modification. (Flag will be set to zero)
History_Receiver.State	1	Next phase
History_Receiver.State0_MTrig	0	This math will not be executed again

6.3.5.2 Report_receiving 1. Phase (Report_*Receiver.State0_MTrig*)

Report_Pointer	DB_ReceiveReport.Name + "." + DB_ReceiveReport.Variable	Pointer gets correct class variable's name: (Sender's) Name.Variable
@Report_Pointer	DB_ReceiveReport.Value	Value is set to pointed variable
DBC_ReceiveReport.MSelect	NOT DBC_ReceiveReport.MSelect	The report array is selected for modification. (Flag will be set to zero)
Report_Receiver.State	1	Next phase
Report_Receiver.State0_MTrig	0	This math will not be executed again

6.4 Removing old (flag = 0) messages

IWS scheduler task *Remove old messages from the interface database* is configured to remove automatically all the old messages (flag =0) in 23:59:59 (~1 day) intervals. Note that new messages (not read messages) has flag =1, so these won't be removed.

Calendar	23:59:59	DBC_ReceiveEvent.Delete	NOT DBC_ReceiveEvent.Delete
Calendar	23:59:59	DBC_ReceiveReport.Delete	NOT DBC_ReceiveReport.Delete
Calendar	23:59:59	DBC_ReceiveHistory.Delete	NOT DBC_ReceiveHistory.Delete
Calendar	23:59:59	DBC_SendAll.Delete	NOT DBC_SendAll.Delete
Calendar	23:59:59	DBC_SendSingle.Delete	NOT DBC_SendSingle.Delete

6.5 Watchdog timer

DB2SMS sends SMS message to the defined phone number if it is doesn't get "heartbeat" signal from IWS. This can happen if IWS program is jammed. IWS updates the interface database's watchdog using *Reset watchdog Counter*-scheduler task.

Clock		00:00:08		DB_Watchdog.Flag	DB_Watchdog.Enabled	
Clock		00:00:08		DB_Watchdog.PRG_Counter	Second	
Clock		00:00:08		DB_Watchdog.Old_PRG_Counter	DB_Watchdog.PRG_Counter - 1	
Clock		00:00:08		DB_Watchdog.A2SMS_Counter	0	
Clock		00:00:08		DBC_Watchdog.Update	NOT DBC_Watchdog.Update	

7 Tags and class tags

Tag	Type	Description
DB_SendAll	Send_All	Send_All –tables active arrays values
DB_SendSingle	Send_Single	Send_One – tables active arrays values
DB_ReceiveEvent	Rec_Event	Received_Events – table's active arrays values
DB_ReceiveHistory	Rec_History	Received_History – table's active arrays values
DB_ReceiveReport	Rec_Report	Received_Report – table's active arrays values
DB_HistoryPool	HisPool	History database's active arrays values
DB_ErrorMessage	String	ErrorMessage – table's active arrays values
DBC_SendAll	ODBC_MsgSender	Send_All –table control (ODBC task)
DBC_SendSingle	ODBC_MsgSender	Send_One – table control (ODBC task)
DBC_ReceiveEvent	ODBC_Receivers	Received_Events – table control (ODBC task)
DBC_ReceiveHistory	ODBC_Receivers	Received_History – table control (ODBC task)
DBC_HistoryPool	ODBC_HisPool	History – database control (ODBC task)
DBC_ReceiveReport	ODBC_Receivers	Received_Report – table control (ODBC task)
DBC_ErrorMessage	ODBC_ErrorMessage	ErrorMessage – table control (ODBC task)
DBC_Send_Phones	ODBC_Phonebook	Phonebook_Sending – table control (ODBC task)
DB_Send_Phones	Phonebook	Phonebook_Sending – table's active arrays values
DBC_Receive_Phones	ODBC_Phonebook	Phonebook_Receiving – table control (ODBC task)
DB_Receive_Phones	Phonebook	Phonebook_Receiving – table's active arrays values
Event_Receiver	Receiver	Event receiver class variables
History_Receiver	Receiver	history receiver class variables
Report_Receiver	Receiver	Report receiver class variables
DBC_Crosslink	ODBC_Phonebook	Report_Crosslinks - table control (ODBC task)
DB_Crosslink	Crosslink	Report_Crosslinks – table's active arrays values
Iface_String0	String	Interface variable
Iface_String1	String	Interface variable
Iface_String2	String	Interface variable
Iface_String3	String	Interface variable
Iface_String4	String	Interface variable
Iface_String5	String	Interface variable
Iface_String6	String	Interface variable
Iface_String8	String	Interface variable
Iface_String7	String	Interface variable
Iface_History_DataGrid_Update	Boolean	History data update after selecting variable name
Iface_History_Grid_Update	Boolean	History variable grid update after selecting plc
Iface_PB_Sending_Grid_Update	Boolean	Outgoing phonebook update after editing it in interface
Iface_Phonebook_receiving_Update	Boolean	PLC data read after selecting PLC from a grid
Send_Phones_Name	String	Selected outgoing phone number's name
AI1Alarm	Boolean	Alarm indication tag for AI1
Report_Pointer	String	Report pointer tag which gets the class variable's name
PLC[10]	PLC	Report variables for PLCs
DBC_Watchdog	ODBC_Watchdog	Watchdog timer's table control (ODBC task)
DB_Watchdog	Watchdog	Watchdog timer – table's active arrays values
WD_ResetTrig	Boolean	Resetting tag for watchdog timer
DB_Config_DSN	String	Interface database DSN
DB_Config_Password	String	Interface database password
DB_Config_Status	String	Interface database status
DB_Config_Username	String	Interface database username
HisPool_DSN	String	History database DSN
HisPool_Password	String	History database password
HisPool_Username	String	History database username

Class	Tag (member)	Type	Description
Crosslink			Crosslink class variable
	ID	Integer	Index
	Name	String	Name
HisPool			History data class variable (history data saving)
	Value	String	
	TimeStamp	String	
	TimeStamp_ms	Integer	milliseconds
	Variable	String	
ODBC_ErrorMessage			Error message class variable
	Select	Boolean	Select trigger
ODBC_HisPool			History data ODBC control class variable
	Select	Boolean	Select trigger
	Insert	Boolean	Insert trigger
	Next	Boolean	Next trigger
ODBC_MsgSender			Message sender class variable (both Group and single)
	Insert	Boolean	Insert trigger
	Select	Boolean	Select trigger
	Delete	Boolean	Delete trigger
ODBC_Phonebook			Phonebook data ODBC control class variable
	Delete	Boolean	Delete trigger
	Insert	Boolean	Insert trigger
	Select	Boolean	Select trigger
	Complete	Boolean	Transaction Complete
ODBC_Receivers			Reveiving message ODBC control class variable
	Status0	Integer	Status
	Select	Boolean	Select trigger
	Next	Boolean	Next trigger
	MSelect	Boolean	Select trigger (DB modification)
	MUpdate	Boolean	Update trigger (DB modification)
	Status1	Integer	Status (DB modification)
	Delete	Boolean	Delete trigger
Phonebook			Phonebook class variable
	Phone	String	Phone number
	Enabled	Boolean	Active flag
	Name	String	Name
PLC			PLC class variable (example)
	AI0	Integer	Analog input 0
	AI1	Integer	Analog input 1
	AI2	Integer	Analog input 2
	AI3	Integer	Analog input 3
	WM0	Integer	Word Memory 0
	WM1	Integer	Word Memory 1
	WM2	Integer	Word Memory 2
	WM3	Integer	Word Memory 3
	Temperature	Real	Scaled temperature
Rec_Event			Receiving event messages class variable
	Flag	Integer	”New message” –flag
	Date	String	date
	Time	String	time
	Phone	String	Phone number
	Message	String	message
	Name	String	sender’s name (read from phonebook by DB2SMS)
	Index	Integer	Database index of the received message
Rec_History			Receiving history messages class variable
	Flag	Integer	”New message” –flag
	Date	String	date
	Time	String	time
	Phone	String	Phone number
	Variable	String	Variable name
	Value	String	value

	Name	String	sender's name (read from phonebook by DB2SMS)
	Index	Integer	Database index of the received message
Rec_Report		Integer	Receiving report messages class variable
	Flag	String	"New message" –flag
	Date	String	date
	Time	String	time
	Phone	String	Phone number
	Variable	String	Variable name
	Value	String	value
	Name	String	sender's name (read from phonebook by DB2SMS)
	Index	Integer	Database index of the received message
Receiver			Message receiver math control class variable
	Active	Boolean	Receiver is active
	Begin_MTrig	Boolean	Beginning of receiving math trigger
	State0_MTrig	Boolean	1. phase math trigger
	State1_MTrig	Boolean	2. phase math trigger
	State2_MTrig	Boolean	3. phase math trigger
	Finish_MTrig	Boolean	Finishing math trigger
	State	Integer	Receiving state
Send_All			Send_All – table ODBC control class variable
	Flag	Integer	"send message" -flag
	Message	String	Message
Send_Single			Send_One –taulun muokkaamista varten
	Flag	Integer	"send message" -flag
	Phone	String	Phone number
	Message	String	Message
ODBC_Watchdog			Watchdog timer ODBC control class variable
	Select	Boolean	Select trigger
	Update	Boolean	Update trigger
Watchdog			Watchdog timer class variable
	Name	String	Watchdog name
	Enabled	Integer	activation
	Flag	Integer	message send (0=send, 1=not send yet)
	PRG_Counter	Integer	This should be changed
	Old_PRG_Counter	Integer	Old value
	A2SMS_Counter	Integer	Error status counter
	Counter_Max	Integer	Error status timeout
	Phone	String	Phone number
	Message	String	Message

8 Known problems

- The IWS MyTest is consuming CPU time so if you are using old PC (e.g. tested with 400MHz, 196MB, it can be very slow. Tests with PC (700MHz, 256MB) was showing that it worked much better. Please report from any problems you have to FF-Automation.
- The program is still under beta testing. Please report from any problems you have to FF-Automation.
- Send message's to

info@ff-automation.com or
antti.moijanen@ff-automation.com